Week 5 :
b. Animation

Anastasia.Bezerianos@lri.fr

(part of this class is based on previous classes from J.Garcia)

Animation

Used to draw images/objects that vary over time

Use the class Animation of JavaFX, main subclasses: Transitions and Timeline Animations

Animations

all Animations have:

rate – speed and direction

cycleCount – the number of animation cycles (positive number) or Animation.INDEFINITE for infinite cycles

autoReverse – inverse directions (false by default)

SetOnFinished - action that will be executed at the end of the animation

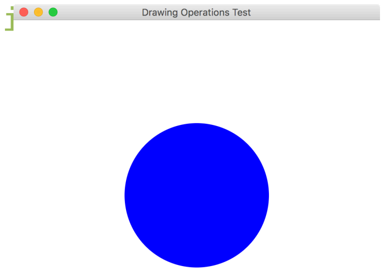
status – can be RUNNING, PAUSED, or STOPPED

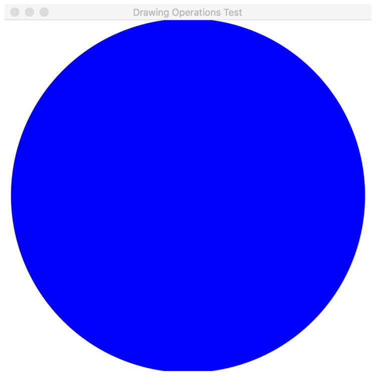
currentTime – time since the start of the last animation cycle

Transition example

```
public class SimpleAnimation extends Application {

…

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Drawing Operations Test");

        FlowPane root = new FlowPane();
        root.setAlignment(Pos.CENTER);

        Ellipse el1 = new Ellipse(10, 10);
        el1.setFill(Color.BLUE);
        root.getChildren().addAll(el1);


        ScaleTransition st = new ScaleTransition(Duration.millis(3000), el1);
        st.setFromX(1); // original x
        st.setFromY(1); // original y
        st.setToX(25);  // final x is 25 times the original
        st.setToY(25);  // final y is 25 times the origi
        st.setCycleCount(Timeline.INDEFINITE);
        st.setAutoReverse(true);
        st.play();

        primaryStage.setWidth(500);
        primaryStage.setHeight(500);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }
```

Animation object lasting 3sec
Each step creates an ActionListener
event (calling MyTimerActionListener)





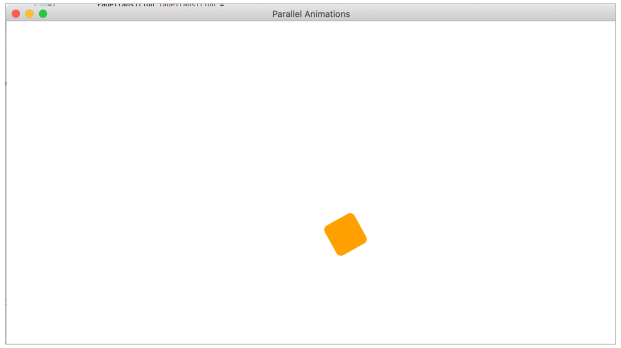
Transitions

Transitions are a subclass of Animation. They:

Incorporate animations in an internal timeline

Can be composed to create multiple animations that are executed in parallel or sequentially

Parallel Transitions



```
public class ParallelTransitionExample extends Application {
     …
    @Override
    public void start(Stage primaryStage) {
        …
        Rectangle rectParallel = new Rectangle(10,200,50, 50);
        rectParallel.setFill(Color.ORANGE);

        FadeTransition fadeTransition =
                new FadeTransition(Duration.millis(3000), rectParallel);
        …
        TranslateTransition translateTransition =
                new TranslateTransition(Duration.millis(2000), rectParallel);
        …
        RotateTransition rotateTransition =
                new RotateTransition(Duration.millis(3000), rectParallel);
        …
        ScaleTransition scaleTransition =
                new ScaleTransition(Duration.millis(2000), rectParallel);
        …

        ParallelTransition parallelTransition = new ParallelTransition(
                    fadeTransition,
                    translateTransition,
                    rotateTransition,
                    scaleTransition
            );
        parallelTransition.setCycleCount(Timeline.INDEFINITE);
        parallelTransition.play();
       …
}
```

What will happen if we replace ParallelTransition with SequentialTransition?

Possible Transitions

FadeTransition (all Nodes)

FillTransition (for Shape nodes, changes fill color)

StrokeTransition (idem for stroke color)

ScaleTransition

TranslateTransition

RotateTransition

PathTransition (Node follows a curved path)

PauseTransition (does nothing, useful in parallel or sequential transitions)

Transitions

The Interpolator property controls the animation acceleration:

Interpolator.LINEAR – constant speed

Interpolator.DISCRETE – a jump from beginning to end

Interpolator.EASE IN – slow at the beginning

Interpolator.EASE OUT – slow at the end

Interpolator.EASE BOTH – ease in and out

Animation Control

For both Transition and Timeline (next)

play() - execute the animation from current position

playFrom() - executer the animation from a specific position

playFromStart() - restart animation

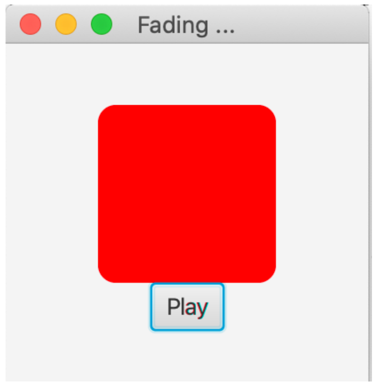
pause() - keeps current position

stop() - stops and resets the start of the animation

jumpTo() – jumps to a specific position

Transition control example

```java
public class ControlAnimationExample extends Application {
    @Override
    public void start(Stage primaryStage) {
        …

        Rectangle rect1 = new Rectangle(10, 10, 100, 100);
        rect1.setFill(Color.RED);
        root.getChildren().addAll(rect1);

        FadeTransition ft = new FadeTransition(Duration.millis(3000), rect1);
        ft.setFromValue(1.0);
        ft.setToValue(0.1);
        ft.setCycleCount(Timeline.INDEFINITE);
        ft.setAutoReverse(true);

        ToggleButton b = new ToggleButton("Play");
        b.setOnAction(e-> {
                if( !b.isSelected())
                {
                    ft.pause();
                    b.setText("Play");
                }
                else {
                    ft.play();
                    b.setText("Stop");
                }
          });
        root.getChildren().add(b);

        primaryStage.setWidth(500); primaryStage.setHeight(500);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }
…
```



Timeline

supports key frame animation

state transitions declared by snapshots
(key frames) at certain times

```
Timeline timeline = new Timeline();
timeline.setCycleCount(Timeline.INDEFINITE);
timeline.setAutoReverse(true);
final KeyValue kv1 = new KeyValue(rectBasicTimeline.xProperty(), 300);
final KeyFrame kf1 = new KeyFrame(Duration.millis(500), kv1);
final KeyValue kv2 = new KeyValue(rectBasicTimeline.xProperty(), 700);
final KeyFrame kf2 = new KeyFrame(Duration.millis(600), kv2);
timeline.getKeyFrames().addAll(kf1,kf2);
timeline.play();
```

Timeline

supports key frame animation

state transitions declared by snapshots

(key frames) at certain times

so very customizable, but …

need to add by hand many keyframes

(see sample code for example)

Timeline

supports key frame animation

state transitions declared by snapshots
(key frames) at certain times

A key frame is defined by the classes:

- a KeyValue (what is animated)
- a KeyFrame (timing)

A Timer using Timeline

```
public class TimerTimeline extends Application {

    // private class constant and some variables
    private static final Integer STARTTIME = 15;
    private Timeline timeline;
    // Make timeSeconds a Property
    private IntegerProperty timeSeconds = new SimpleIntegerProperty(STARTTIME);

    @Override
    public void start(Stage primaryStage) {

        // Setup the Stage and the Scene (the scene graph)
        primaryStage.setTitle(“Timer");
        Group root = new Group();
        Scene scene = new Scene(root, 80, 30);

        // Bind the timerLabel text property to the timeSeconds property
        Label timerLabel = new Label();
        timerLabel.textProperty().bind(timeSeconds.asString());


        // a Button that controls the timer
        Button button = new Button();
        button.setText("start");
        button.setOnAction(e-> {
                if (timeline != null) {
                    timeline.stop();
                }
                timeSeconds.set(STARTTIME);
                timeline = new Timeline();
                timeline.getKeyFrames().add(
                        new KeyFrame(Duration.seconds(STARTTIME+1),
                        new KeyValue(timeSeconds, 0)));
                timeline.playFromStart();
        });

        // layout
        FlowPane panel = new FlowPane();
        panel.getChildren().addAll(button, timerLabel);
        root.getChildren().add(panel);

        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

A Timer using Timeline

```java
public class TimerTimeline extends Application {

    // private class constant and some variables
    private static final Integer STARTTIME = 15;
    private Timeline timeline;
    // Make timeSeconds a Property
    private IntegerProperty timeSeconds = new SimpleIntegerProperty(STARTTIME);

    @Override
    public void start(Stage primaryStage) {

        // Setup the Stage and the Scene (the scene graph)
        primaryStage.setTitle("Timer");
        Group root = new Group();
        Scene scene = new Scene(root, 80, 30);

        // Bind the timerLabel text property to the timeSeconds property
        Label timerLabel = new Label();
        timerLabel.textProperty().bind(timeSeconds.asString());


        // a Button that controls the timer
        Button button = new Button();
        button.setText("start");
        button.setOnAction(e-> {
                if (timeline != null) {
                    timeline.stop();
                }
                timeSeconds.set(STARTTIME);
                timeline = new Timeline();
                timeline.getKeyFrames().add(
                        new KeyFrame(Duration.seconds(STARTTIME+1),
                        new KeyValue(timeSeconds, 0)));
                timeline.playFromStart();
        });

        // layout
        FlowPane panel = new FlowPane();
        panel.getChildren().addAll(button, timerLabel);
        root.getChildren().add(panel);

        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

Chose a starting time

Create a Property that you can then bind to UI elements

Bind your label to the Property

Create a Timeline animation every time you press on the start Button

Add a keyframe every second

more on JavaFX animations

https://docs.oracle.com/javafx/2/animations/jfxpub-animations.htm
https://www.genuinecoder.com/javafx-animation-tutorial/

16